IRISA Participation in JRS 2012 Data-Mining Challenge: Lazy-Learning with Vectorization

Vincent Claveau

IRISA Participation in JRS 2012 Data-Mining Challenge: Lazy-Learning with Vectorization

Vincent Claveau

IRISA – CNRS
Campus de Beaulieu, 35042 Rennes, France
vincent.claveau@irisa.fr

Abstract. In this article, we report on our participation in the JRS Data-Mining Challenge. The approach used by our system is a lazy-learning one, based on a simple k-nearest-neighbors technique. We more specifically addressed this challenge as an opportunity to test Information Retrieval (IR) inspired techniques in such a data-mining framework. In particular, we tested different similarity measures, including one called vectorization that we have proposed and tested in IR and Natural Language Processing frameworks. The resulting system is simple and efficient while offering good performance.

Keywords: Vector space, Vectorization, LSI, k-Nearest Neighbors, Information Retrieval

1 Introduction

This article describes the IRISA participation in the JRS Data-Mining Challenge. The team was composed of Vincent Claveau, IRISA-CNRS, and was identified as vclaveau. The approach used by our system is a lazy-learning one, relying on a k-nearest-neighbors technique (kNN). In this standard data-mining technique, the object to classify is compared with those from the training set. The closest ones then vote for the classes and the final class(es) are attributed to the new object based on these votes.

Such a technique necessitates to define at least two components: how to compute the similarity between a new object and the training set ones, and how to combine the votes to assign the classes to the new objects. For these two components, we used techniques initially developed in the Information Retrieval (IR) domain. In particular, we show that the similarity measure that we have developed, called vectorization, yields better results than usual similarity measures. It implements a second-order distance based on the use of pivots to build a new vector space representation.

The resulting system does not need any learning step *per se* and is very fast: the full processing (from the processing of the training set to the generation of the results for the test set) takes approximately 5s on a laptop computer. The best score that was obtained during the official evaluation is 0.500. It is ranked

17th on the leaderboard, with score only 0.02947 points worse than the best score.

The paper is organized as follows: the next section gives some background on how usual techniques from IR can be used to implement kNN. Section 3 presents our vectorization approach to compute second-order similarities. Section 4 details how the classes are predicted from the nearest-neighbors found. The results obtained by our approach and different variants are presented in Section 5, and some conclusive remarks are given in the last section.

2 First order similarity for nearest neighbors

As we previously said, our whole approach is based on techniques initially developed for Information Retrieval (IR). Thus, we make an analogy between the JRS Challenge data and IR, and more precisely with the vector space model classically used in IR: a vector represents a document (noted d hereafter), and each dimension represents a word. As in IR, instead of computing a distance or similarity directly from the initial vectors, we apply a weighting scheme to the data. This common approach is explained in the two following subsections and is what we call a first order similarity.

2.1 Weighting schemes

Several weighting schemes have been proposed in IR. Their goal is to give more importance to representative attributes/words of an object/document. The TF-IDF is certainly the most well known of these weighting schemes. It is based on considerations developed in several seminal papers [13, 14] and is usually defined as:

$$w_{TF-IDF}(t,d) = TF(t,d) * IDF(t) = tf(t,d) * \log(N/df(t))$$

where $tf(t,d)$ represents the value of the dimension/word t for the vector/document d, N is the total number of vectors and $df(t)$ is the number of vectors having a non-zero dimension t.

Another weighting scheme has been proved much more efficient than the standard TF-IDF for most IR problems. This scheme, called Okapi-BM25, can be seen as a variant of TF-IDF. Its definition is given in Equation 1; it indicates that the weight of word/dimension t in the document/vector d ($k_1 = 2$ and $b = 0.75$ are constants, dl is the length of document, dl_{avg} the average document length).

$$\begin{aligned} w_{BM25}(t,d) &= TF_{BM25}(t,d) * IDF_{BM25}(t) \\ &= \frac{tf(t,d) * (k_1+1)}{tf(t,d) + k_1 * (1 - b + b * dl(d)/dl_{avg})} * \log \frac{N - df(t) + 0.5}{df(t) + 0.5} \end{aligned} \quad , \tag{1}$$

The TF_{BM25} part was initially derived from a probabilistic model of the frequency of terms (dimensions) in the documents (vectors), namely the 2-Poisson

model of Harter [15]. This model represents the distribution of terms in the documents as a mixture of two Poisson distributions: one represents the frequency of terms relevant to describe the document, while the other represents the frequency of non relevant ones [9]. In practice in IR, this TF formula is considered as better than the original one because it includes a normalization based on the size of the document. The IDF_{BM25} part is also derived from probabilistic considerations [15] and is quite similar to the empirically set up standard IDF formula.

Note that these weighting schemes do not change the sparsity of the vectors. Every 0-component of the vectors keeps its value to 0. This property is generally exploited to perform very efficient similarity computations (see below).

2.2 Similarity

The data were provided in a vector representation. Each object is thus described as a vector of 25000 dimensions which is very sparse. This sparse vector representation is very often used in IR, and measures like Minkowsky Lp distances are commonly used to compute similarity between two such vectors. For two vectors x and y, Minkowsky distances are defined by equation 2; p is usually chosen as 1 (Manhattan distance), 2 (Euclidean distance) or ∞ (Chebyshev distance), if $p < 1$, Lp is no longer a distance.

$$Lp(x,y) = \sqrt[p]{\sum_i |x_i - y_i|^p} \quad (2) \qquad cos(x,y) = \frac{\sum_i x_i \cdot y_i}{\|x\| \cdot \|y\|} \quad (3)$$

The cosine similarity (eqn 3) is also very often used in IR and data-mining. Since it is based on the scalar-product of the two vectors, it allows a very efficient computation for sparse vectors since only the components which have non-zero values in both vectors have to be considered. Note that the cosine is equivalent to (i.e. yields the same ordering of neighbours as) the L2 distance if the vectors are normalized: $L2(x,y) = \sqrt{2 - 2 * cos(x,y)}$.

In practice, such distances or similarity measures are computed between the weigthed versions of the vectors (TF-IDF, Okapi or others). More precisely, one vector serves as a query, and its nearest neighbors are the vectors having a minimal distance (or maximal similarity) with it. In IR, it is usual to adopt different weighting schemes for the query vector and the vectors from the collection (training vectors), since the query have some particularities one may want to take into account (for instance, queries in a search engine are often composed of only 2 or 3 words and thus results in a vector much sparser than the text collection ones).

3 Vectorization: second-order similarity

3.1 Principle

However, we have developed a more effective similarity technique, based on a transformation on the initial vector space into another. This transformation,

called Vectorization, has been used in various IR and Text Mining tasks [3] where it has shown to provide both a low complexity and accurate results. As any embedding technique, Vectorization aims to project any similarity computation between two objects in a vector space. Its principle is relatively simple. For each document of the considered collection, it consists of computing using an initial similarity measure (e.g., a standard similarity measure like the cosine), some proximity scores to m pivot-objects. These m scores are then gathered into a m-dimensional vector representing the object, as shown in Figure 1.

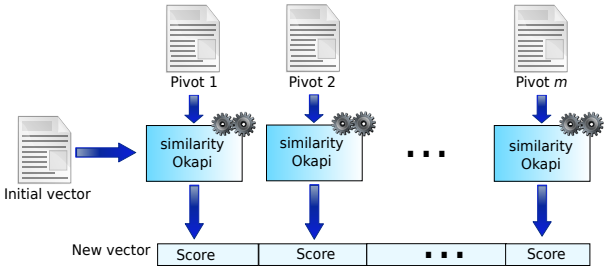

Fig. 1. Vectorization embedding of an example

It is important to notice that the vectorization process changes the representation space. It is not only a space reduction or an approximation of the initial distance as proposed for instance by some authors [2]. It is not either an othogonalization or a linear transform as in Latent semantic Indexing/Analysis (LSI/LSA), probabilistic Latent Semantic analysis (pLSA) [10], Latent Dirichlet Allocation (LDA) [7], principal component analysis (PCA) [1], LDA [6] and even random linear transformations [17]. In these representations, a new vector space is also built, but its dimensions are simply linear combinations of the initial ones.

Changing the representation space, based on the similarity to the pivots, brings up two important properties. Firstly, this embedding helps to reduce the complexity when the initial similarity measure is too expensive to be used inline [4]. Of course, this property is not really useful in the context of this JRS Challenge but appears as important when vectorization is used for Information Retrieval. Secondly, vectorization will consider two objects as close if they have the same behaviour regarding the pivots, that they are close to same pivots and far for the same pivots. As with LSI or LDA, this indirect comparison makes it possible to pair two objects even if they do not bear common components in the initial vector space.

3.2 Pivots

The pivots can be any objects provided that we are able to compute a similarity between them and the initial vector. They may be artificially created or generated from existing data. In this JRS Challenge context, we have adopted the latter solution: we have built 83 pivots, one per class. Each pivot is simply a sum

of a random selection of the vectors belonging to the corresponding class. Each training object is then represented in this 83-dimension space by computing its distance (we used a cosine) to each of the 83 pivots. This embedding provides a more robust representation than the initial 25000-dimension one; indeed, it allows to consider two objects as close, because they are close from the same pivots, even if they were not considered as close in the initial space. This property may be important in the JRS Challenge context in which a topic could be expressed by different MeSH term combinations.

3.3 Similarity

Comparing two 'vectorized' objects can be performed in a very standard way in the new vector space with a L2 distance for example. Yet, it is important to note one property: as we previously underlined it, computing L2 or cosine between two vectors can be very efficiently done when the vectors are sparse since it needs to consider only the dimensions having a non-zero value in each vector. When the vector space is not sparse and has high dimensionality, the cost of computing the distance can be very important.

Yet, many algorithms are available to compute or approximate very efficiently such distances. To save processing time, these techniques either address the completeness of the search, or the accuracy of distance calculation. For instance, the *hashing*-based techniques [16, 5] tackle the completeness: the space is divided into portions, and the search is conducted on a subset of these portions. The NV-tree [12] pushes this approach further as it also approximates L2 distances of the portion chosen. Finally, it provides results in $\mathcal{O}(1)$ (ie, a constant time based on a single disk access), whatever the number of vectors in the space. For the experiments presented below and given the small number of pivots and training vectors, such techniques were not necessary; a direct cosine (i.e. equivalent to a direct L2) computation was performed.

4 Vote

Based on the similarity measure described above, the nearest-neighbors of an object can be efficiently retrieved. For each of them, we have a list of its classes and its similarity score. To gain more robustness, we ran different runs. Since the pivots are randomly generated, the results obtained varied from one run to another. Here again, to combine the results, we made an analogy to the IR process. More precisely, we made an analogy to meta-search whose goal is to combine the ranked results of multiple systems. Thus we used one of the most-know combination formula used for meta-search, namely CombMNZ [8, 11]. CombMNZ re-orders the classes of the neighbors retrieved by all the systems. Let us note Sys_c the set of systems (or runs in our case) proposing c as a possible class (that is, with a non-zero score) for the considered test object; the CombMNZ score is then defined as:

$$CombMNZ(c) = \sum_{s \in \text{Sys}_c} score(s, i) * |\text{Sys}_c| \quad (4)$$

It is based on score ($score(s, i)$) associated with each class. In IR, this score is the similarity between the query and the document considered. In our case, it is the sum of similarities with close neighbors belonging to class c as obtained by vectorization. The full process is schematized in Figure 2 The resulting similarity list was cut based on a fixed threshold on the CombMNZ score.

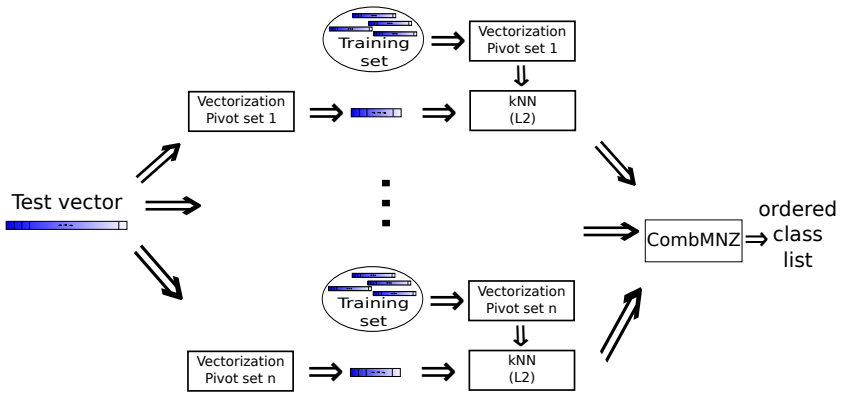

Fig. 2. Complete voting process based on CombMNZ

5 Results

In Table 1, we present different results obtained with the system described in the previous sections. We indicate the results obtained when using different similarity measures instead of vectorization. We thus report the scores obtained by first-order similarities like TF-IDF/L2 and Okapi, as well as the transformation-based similarities LSI and LDA with several sizes of space. These results are expressed in terms of f-measure, as defined by the organisers, and we also indicate the optimal f-measure, that is, the best f-measure that could be obtained if the class list produced by our systems (i.e. by CombMNZ) would have been cut at the best place. In order to have a precise estimate of these measures, we use a 20-fold cross-validation.

Several points are worth noting. Firstly, it seems that the weighting schemes have almost no influence on the results. This result is surprising but difficult to interpret given that no information was given on how the vector values were computed and what they represent.

Another interesting fact is that systems based on a dimensionality reduction, such as LSI, LDA or vectorization, perform better on average than those relying on the initial highly dimensional vector space. Here again, given that no information is given on which dimension represents which MeSH term and how the hierarchical nature of the MeSH was taken into account, these results are

	f-measure	optimal f-measure
no weighting/L2	0.4305	0.5992
TF-IDF/L2	0.4464	0.6023
Okapi	0.4529	0.6142
LSI 83 dims	0.4605	0.6253
LSI 150 dims	0.4795	0.6397
LSI 250 dims	0.4811	0.6435
LSI 350 dims	0.4815	0.6497
LDA 83 dims	0.4087	0.5801
LDA 150 dims	0.4384	0.6057
LDA 250 dims	0.4514	0.6122
LDA 350 dims	0.4482	0.6101
Vectorization 83 pivots	0.5106	0.6915

Table 1. F-measure and optimal f-measure for different IR-inspired kNN systems

difficult to interpret. But, since the 25,000 MeSH index terms used to build the vectors are not independent, it can be supposed that the space reduction helps to match vectors belonging to the same categories even if they are described by different (but dependent) terms.
Finally, among all tested similarity measures, Vectorization performs best. Moreover, the optimal results obtained needs less dimensions than the LSI or LDA techniques, resulting more compact vectors. Several other experiments, not reported here, have been conducted to assess the influence of other parameters. They showed that the number k of neighbors has only a small effect on the results. There is almost no difference for k varying between 3 and 20. Also, different aggregating techniques have been tested beside CombMNZ, such as CombSUM, CombMAX, Condoret vote... All of them yielded lower results, as it has been verified in many IR tasks.

Last, let us note that the official score obtained by our system, computed on the final test data, is 0.50632. It is ranked 17^{th} on the leaderboard, with score only 0.02947 points worse than the best performing system.

6 Conclusions

The approach that we proposed for this JRS Data-Mining Challenge is efficient and yields good results. One of its main characteristics is that it does not rely on a complex Machine Learning approach; it rather uses a lazy learning system inspired by Information Retrieval techniques. In particular, this challenge offered us an opportunity to emphasize the interest of using vectorization to build compact yet precise vector representation of the data in this data-mining framework. Thanks to this representation, the resulting system is very fast while yielding good results.

Many improvements could be done in order to achieve better scores. In particular, one remaining problem was to decide where the list of potential classes

provided by CombMNZ should be cut. As we have shown, this choice has a major impact on the results; indeed, if the optimal cut-off value had been chosen for each test object, the results obtained would have reached 0.69. The choice of the evaluation measure (simple f-measure) has made this cut-off step somewhat artificially important regarding the task. Other evaluation measures like the mean average precision (mAP) could have provided a more flexible and informative framework.

References

1. Berry, M., Martin, D.: Principal component analysis for information retrieval. In: Kontoghiorghes, E. (ed.) Handbook of Parallel Computing and Statistics. Statistics: A Series of Textbooks and Monographs (2005)
2. Bourgain, J.: On Lipschitz embedding of finite metric spaces in hilbert space. Israel Journal of Mathematics 52(1) (1985)
3. Claveau, V., Lefvre, S.: Topic segmentation of tv-streams by mathematical morphology and vectorization. In: Procedings of the InterSpeech conference. Florence, Italy (2011)
4. Claveau, V., Tavenard, R., Amsaleg, L.: Vectorisation des processus d'appariement document-requête. In: 7e conférence en recherche d'informations et applications, CORIA'10. pp. 313–324. Sousse, Tunisie (Mar 2010)
5. Datar, M., Immorlica, N., Indyk, P., Mirrokni, V.: Locality-sensitive hashing scheme based on p-stable distributions. In: Proc. of the 20th ACM Symposium on Computational Geometry. Brooklyn, New York, USA (2004)
6. David M. Blei, Andrew Y. Ng, M.I.J.: Latent dirichlet allocation. Journal of Machine Learning Research 3(4-5), 993–1022 (2003)
7. Dumais, S.: Latent semantic analysis. ARIST Review of Information Science and Technology 38(4) (2004)
8. Fox, E., Shaw, J.: Combination of multiple searches. In: Proceedings of the 2nd Text Retrieval Conference (TREC-2), NIST Special Publication. pp. 243–252 (1994)
9. Harter, S.: A probabilistic approach to automatic keyword indexing. Journal of the american society for information science 26(6), 197–206 (1975)
10. Hofmann, T.: Probabilistic latent semantic indexing. In: Proc. of SIGIR. Berkeley, USA (1999)
11. Lee, J.: Combining multiple evidence from different properties of weighting schemes. In: Proceedings of the 18th Annual ACM-SIGIR. pp. 180–188 (1995)
12. Lejsek, H., Asmundsson, F., Jónsson, B., Amsaleg, L.: Nv-tree: An efficient disk-based index for approximate search in very large high-dimensional collections. IEEE Trans. on Pattern Analysis and Machine Intelligence 99(1) (2008)
13. Luhn, H.P.: The automatic creation of literature abstracts. IBM Journal on Research and Development 2(2) (1958)
14. Spärck Jones, K.: A statistical interpretation of term specificity and its application in retrieval. Journal of Documentation 28(1) (1972)
15. Spärck Jones, K., Walker, S.G., Robertson, S.E.: Probabilistic model of information retrieval : Development and comparative experiments. Information Processing and Management 36(6) (2000)
16. Stein, B.: Principles of hash-based text retrieval. In: Proc. of SIGIR. Amsterdam, Pays-Bas (2007)
17. Vempala, S.: The Random Projection Method, Discrete Mathematics and Theoretical Computer Science, vol. 65. AMS (2004)